IN THE SPECIFICATION

Please replace the fourth paragraph on page 1 with the following paragraph:

In the following discussion, the term "testing" ~~Atesting@~~ relates to a facility for exercising a component against an expected behaviour. "Verification" ~~Averification@~~ on the other hand is the capability to determine whether the component is valid for execution. References to 3rd Generation Language, or simply 3GL are to conventional compiled programming languages, for example JAVA ~~Java~~, C#, C++ or ADA ~~Ada~~. XML is the abbreviation for eXtensible Mark-up Language.

Please replace the sixth paragraph on page 1 with the following paragraph:

Business procedures can be represented in a number of alternative notations. Known examples of business logic representations include: canonical XML based representations (for example BPML, ebXML, BIZTALK ~~BizTalk~~, XLANG ~~Xlang~~, WS-BPEL or WSFL) and proprietary representations (HIPAA, IAA or FIPA).

Please replace the paragraph that spans pages 1 and 2 with the following paragraph:

In the present invention, the term "contextual information" encompasses information about the architecture and services available on a target platform (the combination of hardware processor and operating system), the preferred native language for the target platform and the capabilities of runtime context available on the target platform. Preferably, runtime context is provided to support a standard environment for executables and thus minimise the amount of source code that has to be generated. Runtime context includes, for example: the available programming environments; the available event systems; and accessibility to local files and services and other dependencies in the target platform, such as third party components or legacy systems. An event system is a system that dispatches events in some form, for example JAVA ~~Java~~ Message Service, JAVABEAN ~~JavaBean~~ Event Listener mechanism, MS WINDOWS ~~Windows~~ or XWINDOWS ~~Xwindows~~ event dispatcher.

Please replace the third full paragraph on page 2 with the following paragraph:

One ambitious example of a distributed processing environment is the GRID. The GRID is a runtime environment that enables processing tasks to be allocated to computing resources, potentially available across an internet or intranet. To a user, the GRID would appear as a large virtual computing system. The GRID facilitates secure, coordinated resource-sharing between individuals and corporations alike. Standards have been defined in the area of Grid computing (Open Grid Services Architecture), and an organisation has been established to help promote these standards, the Globus Alliance ~~see http://www.globus.org/~~.

Please replace the sixth full paragraph on page 4 with the following paragraph:

Advantageously, the executables are generated as source code in a third generation language. The third generation language may be one of the set of languages including C, C++, C#, ADA ~~Ada~~, JAVA ~~Java~~, DELPHI ~~Delphi~~, VISUAL BASIC ~~Visual Basic~~, and FORTRAN 90.

Please replace the first paragraph on page 6 with the following paragraph:

A key concept in the present invention is the concept of autonomic operation. Autonomic computing can be viewed as an approach to self-managed computing systems with a minimum of human interference. By analogy with the human body's ~~body=s~~ autonomic nervous system, an autonomic computing system seeks to control key functionality without "conscious awareness or involvement" ~~Aconscious awareness or involvement@~~. Here it is particularly desirable that the need for a conscious human analyst be minimised as the development process proceeds around a development loop.

Pleaso replace the first full paragraph on page 8 with the following paragraph:

A process calculus defines strongly typed interfaces between processes (known as ports or channels). The definition of a type, in the context of such interfaces, includes both static types, which are typical in conventional programming languages (e.g. JAVA ~~Java~~, C#, C++, etc), as well as "behavioural" types, which are based on the externally observable behaviour of processes. It is the use of a behavioural type that allows the capture of the "semantic"

interfaces between processes and, indeed, their representation as ports or channels. Processes communicate with each other through these ports or channels, thereby interacting in a semantically correct way with one another.

Please replace the paragraph beginning "(ii) Loop constructs" that spans pages 11 and 12 with the following paragraph:

(ii) Loop constructs

"while" ~~>while=~~ loops are supported to enable contained activities to be performed until a specified conditional expression evaluates to false.

Please replace the second full paragraph on page 15 with the following paragraph:

The interaction graph illustrated in Figure 4 starts with the reception of monitoring events 402. Events can be received statically (that is to say, as the result of queries on a historic database of recorded monitoring events) or dynamically in "realtime" ~~Arealtime@~~, while the associated processes are executing.

Please replace the second full paragraph on page 17 with the following paragraph:

An illustrative scenario is shown in Figure 5. The scenario is based on a "central policy maker" 502 who is responsible for determining policy. The business logic of this scenario can be specified in the RIF distributed processing environment. Policy is encoded in a declarative XML form that maps directly to a process specification in RIF, RIF markup language (RIFML) 504. The policy maker 502 delivers the policy to those entities 506A-C that must implement the policy (hereinafter referred to generally as the "constituency"). "Constituents" ~~AConstituents@~~ 506A-C then have a responsibility for adhering to (or implementing) that policy.

Please replace the paragraph that spans pages 17 and 18 with the following paragraph:

The central policy maker's ~~maker=s~~ notion of what the policy is and a constituent's implementation of the same policy may diverge, for example in the area of legacy systems.

Consider the situation where an incorrectly modelled wrapper for a legacy system results in a message exchange that is invalid, where invalid means invalid against the central policy maker's definition of that policy. This might happen due to a timing issue when wrapping a synchronous system into an asynchronous message-passing infrastructure. This would result in monitoring information from the execution of the distributed processes being compared against the original business logic (i.e. the policy as defined by the central policy maker). This would highlight any differences between the required behaviour and the implementation of the policy and so enable the translation mechanism to be autonomically changed so that the correct behaviour can be ensured without the need for a human analyst to amend either business logic representation or executable.